Лабораторная работа 1. Простые модели компьютерной сети

Абакумова Олеся Максимовна, НФИбд-02-22

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Приобретение навыков моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования.

# 2 Теоретическое введение

Network Simulator (NS-2) — один из программных симуляторов моделирования процессов в компьютерных сетях. NS-2 позволяет описать топологию сети, конфигурацию источников и приёмников трафика, параметры соединений (полосу пропускания, задержку, вероятность потерь пакетов и т.д.) и множество других параметров моделируемой системы. Данные о динамике трафика, состоянии соединений и объектов сети, а также информация о работе протоколов фиксируются в генерируемом trace-файле.

NS-2 является объектно-ориентированным программным обеспечением. Его ядро реализовано на языке С++. В качестве интерпретатора используется язык скриптов (сценариев) OTcl (Object oriented Tool Command Language). NS-2 полностью поддерживает иерархию классов С++ и подобную иерархию классов интерпретатора OTcl. Обе иерархии обладают идентичной структурой, т.е. существует однозначное соответствие между классом одной иерархии и таким же классом другой. Объединение для совместного функционирования С++ и OTcl производится при помощи TclCl (Classes Tcl). В случае, если необходимо реализовать какую-либо специфическую функцию, не реализованную в NS-2 на уровне ядра, для этого используется код на С++.

Более подробно про NS-2 см. в [1].

# 3 Выполнение лабораторной работы

## 3.1 Шаблон сценария для NS-2

В своём рабочем каталоге создадим директорию mip, к которой будут выполнять- ся лабораторные работы. Внутри mip создадим директорию lab-ns, а в ней файл shablon.tcl (рис. 1):

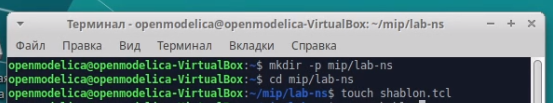


Рис. 1: Создание рабочих директорий и файла

Откроем на редактирование файл shablon.tcl. Можно использовать любой текстовый редактор типа emacs.Я использую nano. Создадим объект типа Simulator, переменную nf и укажем, что требуется открыть на запись nam-файл для регистрации выходных результатов моделирования (рис. 2):

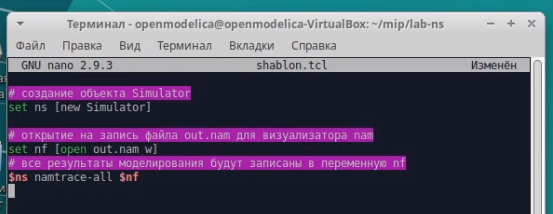


Рис. 2: Создание объекта и переменной вместе с требованием

Вторая строка даёт команду симулятору записывать все данные о динамике модели в файл out.nam.

Далее создадим переменную f и откроем на запись файл трассировки для реги- страции всех событий модели (рис. 3):

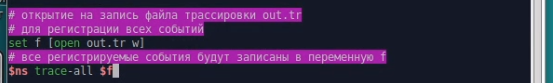


Рис. 3: Создание переменной f и открытие на запись файла трассировки

После этого добавим процедуру finish, которая закрывает файлы трассировки и запускает nam (рис. 4):

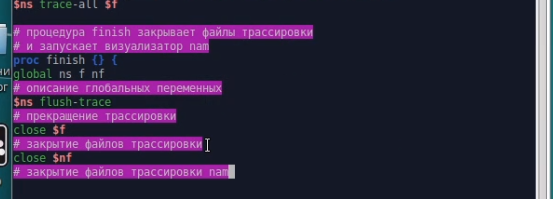


Рис. 4: Добавление процедуры

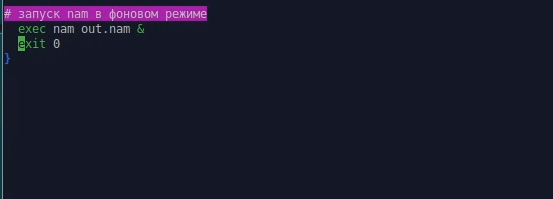


Рис. 5: Добавление процедуры(продолжение)

Наконец, с помощью команды at указываем планировщику событий, что проце- дуру finish следует запустить через 5 с после начала моделирования, после чего запустить симулятор ns (рис. 6):

Рис. 6: Указания для планироващика событий

Рис. 6: Указания для планироващика событий

Сохранив изменения в отредактированном файле shablon.tcl и закрыв его, можно запустить симулятор командой (рис. 7):

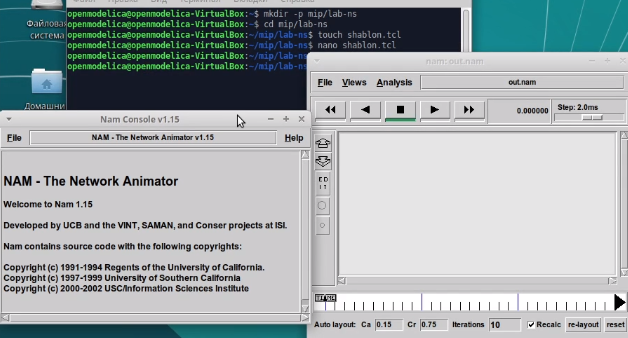


Рис. 7: Запуск симулятора

Получившийся шаблон можно использовать в дальнейшем в большинстве раз- рабатываемых скриптов NS-2, добавляя в него до строки $ns at 5.0 “finish” описание объектов и действий моделируемой системы.

## 3.2 Простой пример описания топологии сети, состоящей из двух узлов и одного соединения

Требуется смоделировать сеть передачи данных, состоящую из двух узлов, соединённых дуплексной линией связи с полосой пропускания 2 Мб/с и задержкой 10 мс, очередью с обслуживанием типа DropTail. От одного узла к другому по протоколу UDP осуществляется передача пакетов, размером 500 байт, с постоянной скоростью 200 пакетов в секунду.

Скопируем содержимое созданного шаблона в новый файл (рис. 8):

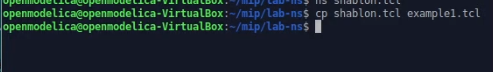


Рис. 8: Копирование шаблона

Открыв example1.tcl на редактирование, добавим в него до строки $ns at 5.0 “finish” описание топологии сети (рис. 9):

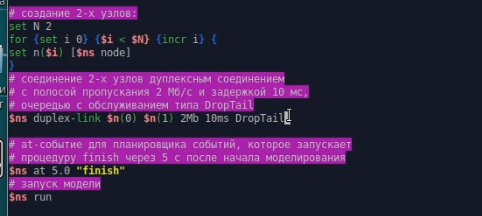


Рис. 9: Добавление строк для описания новой топологии сети

Создадим агенты для генерации и приёма трафика (рис. 10):

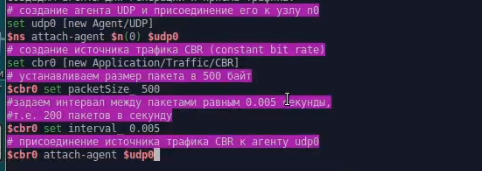


Рис. 10: Создание агентов

Создаётся агент UDP и присоединяется к узлу n0. В узле агент сам не может генерировать трафик, он лишь реализует протоколы и алгоритмы транспортного уровня. Поэтому к агенту присоединяется приложение. В данном случае — это источник с постоянной скоростью (Constant Bit Rate, CBR), который каждые 5 мс посылает пакет R = 500 байт. Таким образом, скорость источника:

Далее создадим Null-агент, который работает как приёмник трафика, и прикрепим его к узлу n1, соединим их между собой (рис. 11):

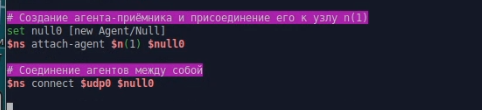


Рис. 11: Создание агента-приемника и соединение между ними

Для запуска и остановки приложения CBR добавляются at-события в планировщик событий (перед командой $ns at 5.0 “finish”) (рис. 12):

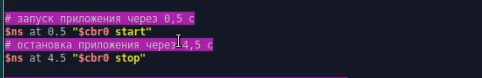


Рис. 12: Добавление at-событий

Сохранив изменения в отредактированном файле и запустив симулятор (рис. 13):

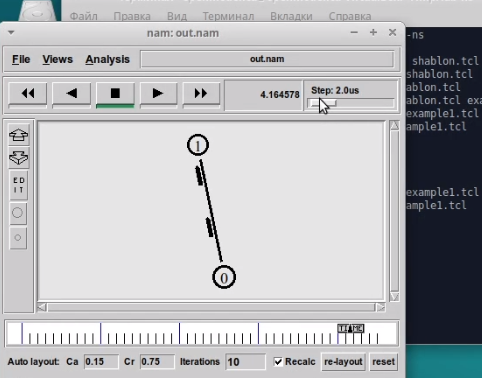


Рис. 13: Запуск example1.tcl

получим в качестве результата запуск аниматора nam в фоновом режиме.

При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1. Это процесс можно замедлить, выбирая шаг отображения в nam. Можно осуществлять наблюдение за отдельным пакетом, щёлкнув по нему в окне nam, а щёлкнув по соединению, можно получить о нем некоторую информацию.

## 3.3 Пример с усложнённой топологией сети

Описание моделируемой сети:

– сеть состоит из 4 узлов (n0, n1, n2, n3);

– между узлами n0 и n2, n1 и n2 установлено дуплексное соединение с пропускной способностью 2 Мбит/с и задержкой 10 мс;

– между узлами n2 и n3 установлено дуплексное соединение с пропускной способностью 1,7 Мбит/с и задержкой 20 мс;

– каждый узел использует очередь с дисциплиной DropTail для накопления пакетов, максимальный размер которой составляет 10;

– TCP-источник на узле n0 подключается к TCP-приёмнику на узле n3 (по-умолчанию, максимальный размер пакета, который TCP-агент может генерировать, равняется 1KByte);

– TCP-приёмник генерирует и отправляет ACK пакеты отправителю и откидывает полученные пакеты;

– UDP-агент, который подсоединён к узлу n1, подключён к null-агенту на узле n3 (null-агент просто откидывает пакеты);

– генераторы трафика ftp и cbr прикреплены к TCP и UDP агентам соответственно;

– генератор cbr генерирует пакеты размером 1 Кбайт со скоростью 1 Мбит/с;

– работа cbr начинается в 0,1 секунду и прекращается в 4,5 секунды, а ftp начинает работать в 1,0 секунду и прекращает в 4,0 секунды.

Скопируем содержимое созданного шаблона в новый файл и откроем example2.tcl на редактирование (рис. 14):

Рис. 14: Создание example2.tcl

Рис. 14: Создание example2.tcl

Создадим 4 узла и 3 дуплексных соединения с указанием направления (рис. 15):

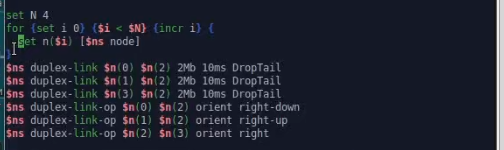


Рис. 15: Создание узлов и соединения

Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP (рис. 16):

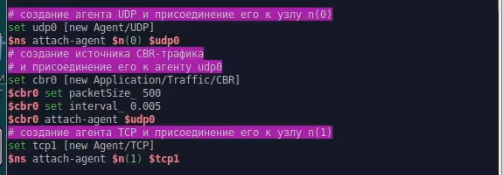


Рис. 16: Создание агента UDP и TCP

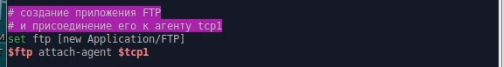


Рис. 17: Создание агента UDP и TCP(продолжение)

Создадим агенты-получатели (рис. 18):

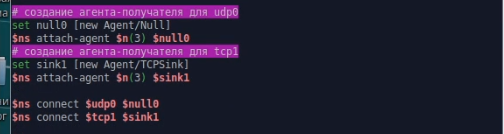


Рис. 18: Создание агентов-получателей

Соединим агенты udp0 и tcp1 и их получателей, описание цвета каждого потока и отслеживание событий в очереди (рис. 19):

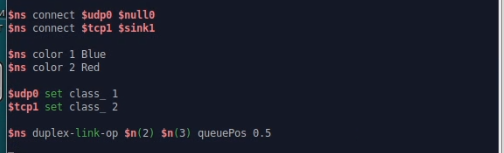


Рис. 19: Соединение агентов и получателей,описание цветов и отслеживание событий

Наложение ограничения на размер очереди и добавление at-событий (рис. 20):



Рис. 20: Наложение ограничения и добавление at-событий

Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования (рис. 21):

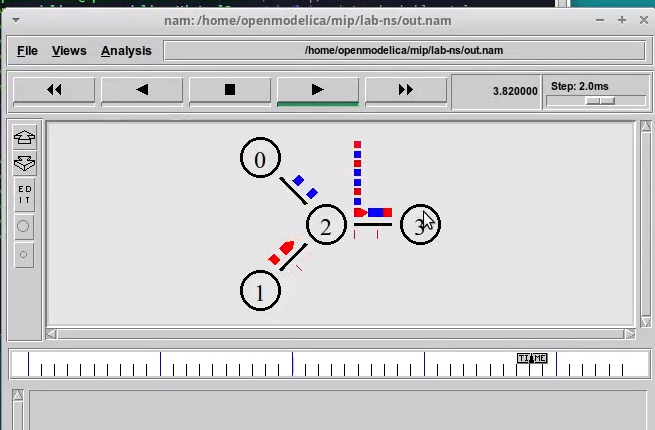


Рис. 21: Запуск example2.tcl

При запуске скрипта можно заметить, что по соединениям между узлами n(0)–n(2) и n(1)–n(2) к узлу n(2) передаётся данных больше, чем способно передаваться по соединению от узла n(2) к узлу n(3). Действительно, мы передаём 200 пакетов в секунду от каждого источника данных в узлах n(0) и n(1), а каждый пакет имеет размер 500 байт. Таким образом, полоса каждого соединения 0, 8 Mb, а суммарная — 1, 6 Mb. Но соединение n(2)–n(3) имеет полосу лишь 1 Mb. Следовательно, часть пакетов должна теряться. В окне аниматора можно видеть пакеты в очереди, а также те пакеты, которые отбрасываются при переполнении.

## 3.4 Пример с кольцевой топологией сети

Требуется построить модель передачи данных по сети с кольцевой топологией и динамической маршрутизацией пакетов:

– сеть состоит из 7 узлов, соединённых в кольцо;

– данные передаются от узла n(0) к узлу n(3) по кратчайшему пути;

– с 1 по 2 секунду модельного времени происходит разрыв соединения между узлами n(1) и n(2);

– при разрыве соединения маршрут передачи данных должен измениться на резерв- ный.

Скопируем содержимое созданного шаблона в новый файл и откроем example3.tcl на редактирование (рис. 22):

Рис. 22: Создание example3.tcl

Рис. 22: Создание example3.tcl

Опишем топологию моделируемой сети и соединим узлы так, чтобы создать круговую топологию (рис. 23):

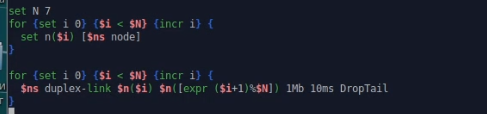


Рис. 23: Создание example3.tcl

Каждый узел, за исключением последнего, соединяется со следующим, последний соединяется с первым. Для этого в цикле использован оператор %, означающий остаток от деления нацело.

Зададим передачу данных от узла n(0) к узлу n(3) (рис. 24):

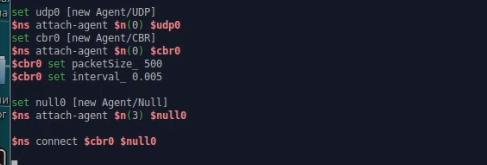


Рис. 24: Задание передачи данных от узла к узлу

Данные передаются по кратчайшему маршруту от узла n(0) к узлу n(3), через узлы n(1) и n(2) (рис. 27).

Добавим команду разрыва соединения между узлами n(1) и n(2) на время в одну секунду, а также время начала и окончания передачи данных (рис. 25):

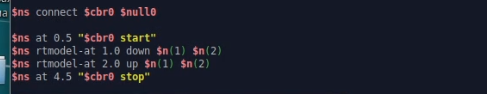


Рис. 25: Добавление команды разрыва и времени начала и окончания для передачи данных

Передача данных при кольцевой топологии сети в случае разрыва соединения (рис. 28).

Добавив в начало скрипта после команды создания объекта Simulator (рис. 26):

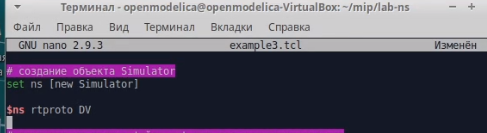


Рис. 26: Добавление строчки в начало

увидим, что сразу после запуска в сети отправляется небольшое количество маленьких пакетов, используемых для обмена информацией, необходимой для марш- рутизации между узлами (рис. 29). Когда соединение будет разорвано, информация о топологии будет обновлена, и пакеты будут отсылаться по новому маршруту через узлы n(6), n(5) и n(4).

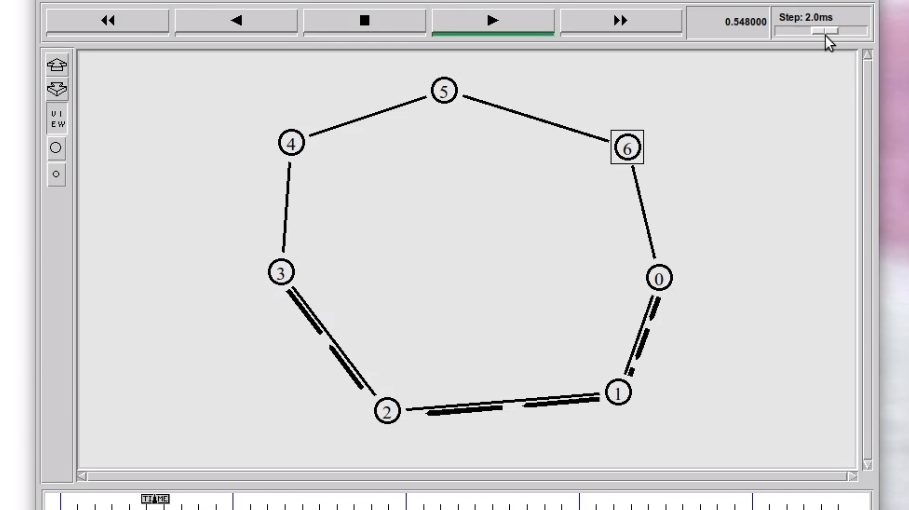


Рис. 27: Передача данных по кратчайшему пути сети с кольцевой топологией

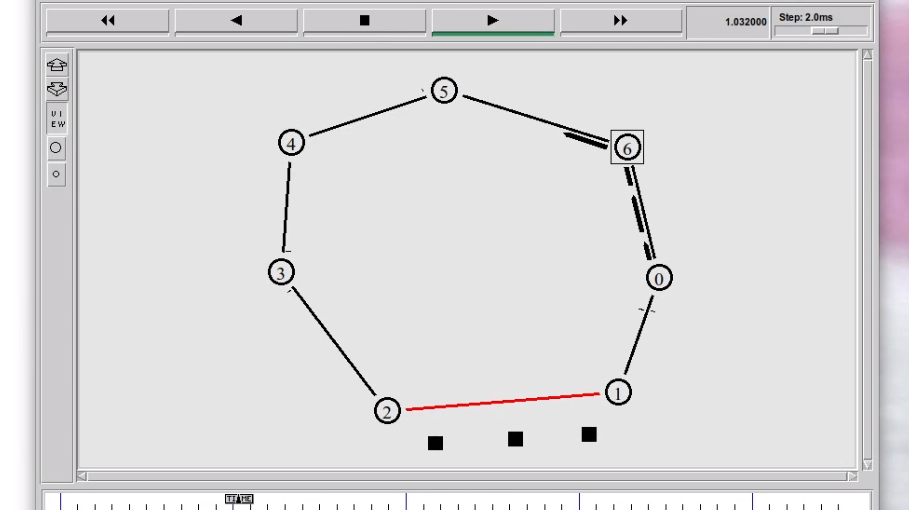


Рис. 28: Передача данных по сети с кольцевой топологией в случае разрыва соединения

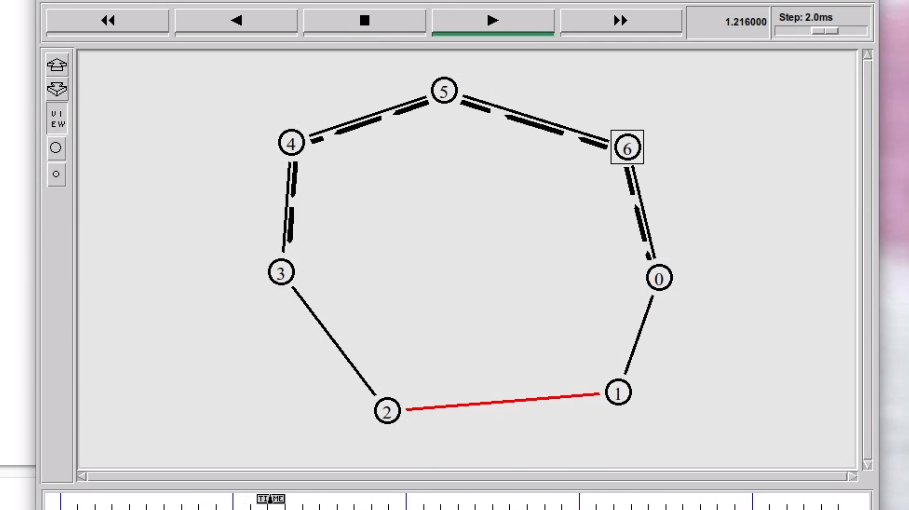


Рис. 29: Маршрутизация данных по сети с кольцевой топологией в случае разрыва соединения

## 3.5 Упражнение

Внесем следующие изменения в реализацию примера с кольцевой топологией сети: – топология сети должна соответствовать представленной на рисунке (рис. 31);

– передача данных должна осуществляться от узла n(0) до узла n(5) по кратчайшему пути в течение 5 секунд модельного времени (рис. 32);

– передача данных должна идти по протоколу TCP (тип Newreno), на принимаю- щей стороне используется TCPSink-объект типа DelAck; поверх TCP работает протокол FTP с 0,5 до 4,5 секунд модельного времени;

– с 1 по 2 секунду модельного времени происходит разрыв соединения между узлами n(0) и n(1) (рис. 33);

– при разрыве соединения маршрут передачи данных должен измениться на ре- зервный (рис. 34), после восстановления соединения пакеты снова должны пойти по кратчайшему пути (рис. 35).

Реализация:

Скопировав шаблон в новый файл example4.tcl и открыв его на редактировани,пропишем следующее (рис. 30):



Рис. 30: Содержание файла example4.tcl

При запуске симуляции,мы получаем кольцевую топологию вида (рис. 31):

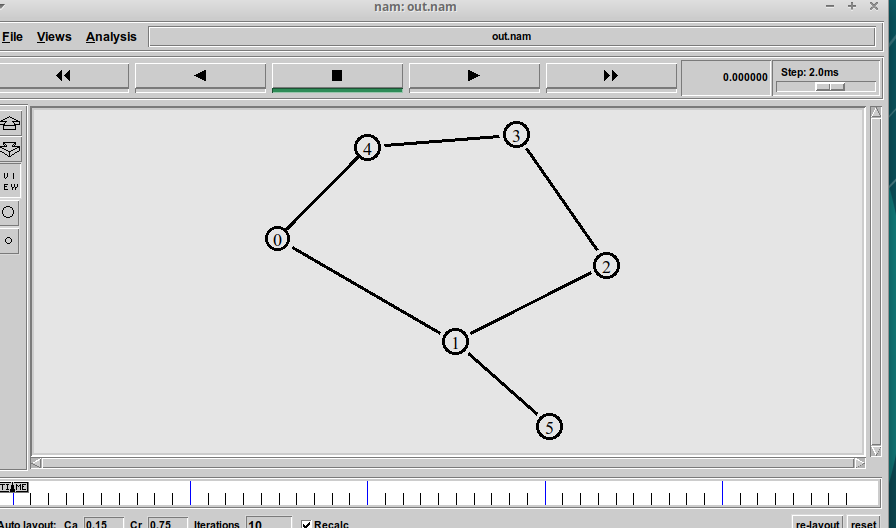


Рис. 31: Пример кольцевой топологии

Передача данных от узла n(0) до узла n(5) осуществялется по кратчайшему пути (рис. 32):

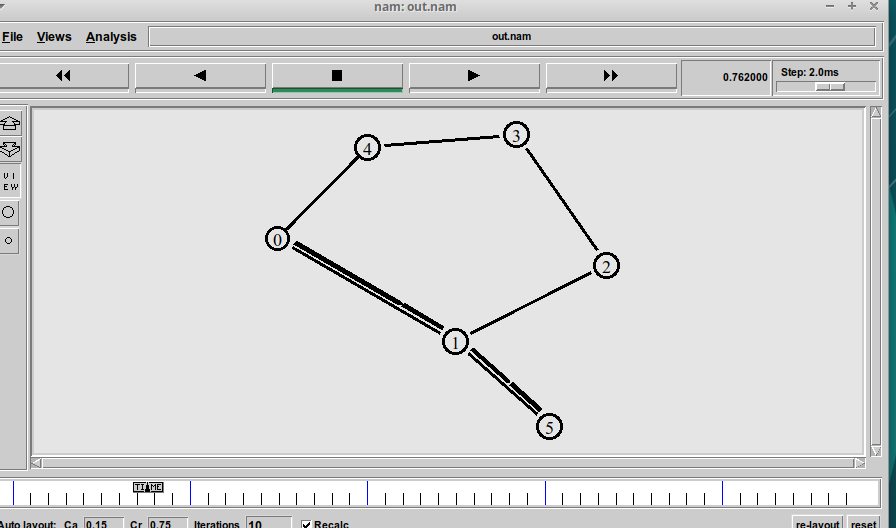


Рис. 32: Путь передачи данных

Далее происходит разрыв соединения между узлами n(0) и n(1) с 1 по 2 секунду моделного времени (рис. 33):

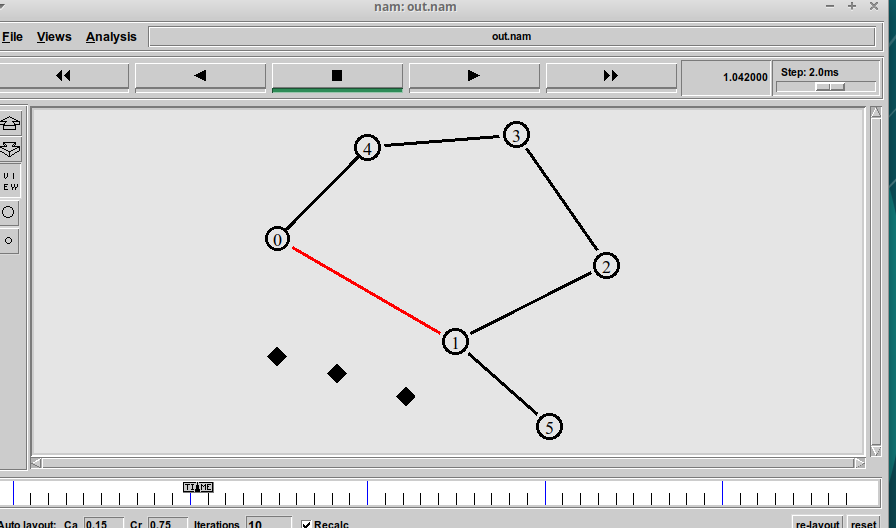


Рис. 33: Разрыв между узлами

После разрыва соединения, маршрут перестроен на резервный (рис. 34):

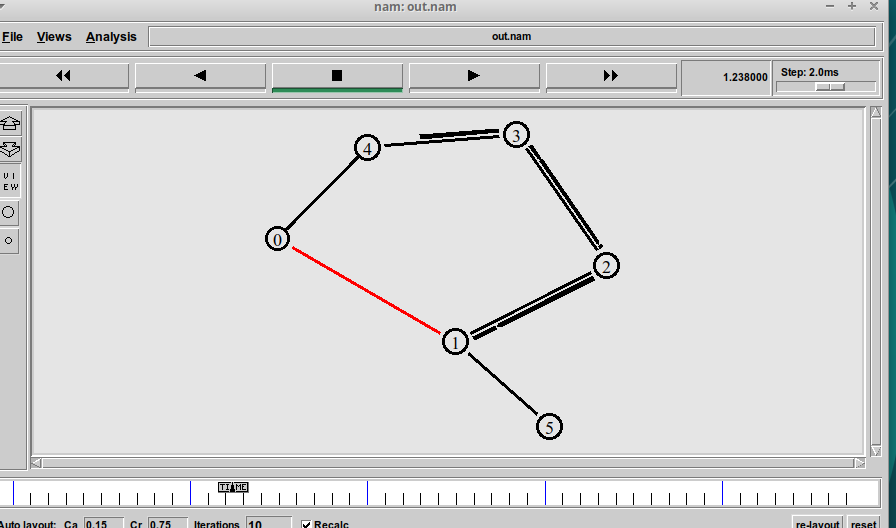


Рис. 34: Резервный маршрут

После восстановления соединения, пакеты снова пошла по кратчайшему пути (рис. 35):

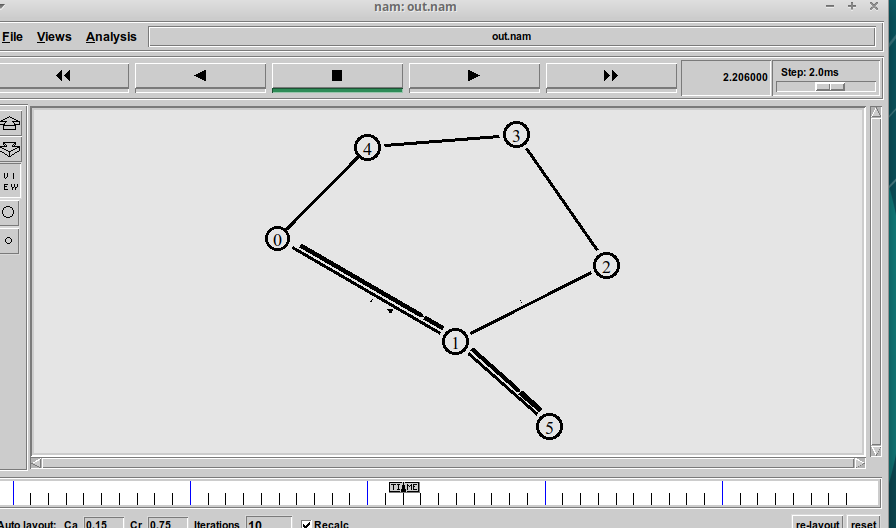


Рис. 35: Восстановленный маршрут передачи пакетов

# 4 Выводы

Во время выполнения данной лабораторной работы я приобрела навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также провела анализ полученных результатов моделирования.

# Список литературы

1. Королькова А.В., Кулябов Д.С. Моделирование информационных процессов. МСК.: Российский университет дружбы народов, 2014. 191 с.